T.S

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/557,633	04/25/2000	Leona Dryden Baumgart	ST9-97-054	9263

7590 12/02/2003

David Victor Esq
315 South Beverly Drive
Suite 210
Beverly Hills, CA 90212

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	11

DATE MAILED: 12/02/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

T5

Office Action Summary	Application No.	Applicant(s)	
	09/557,633	BAUMGART ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09/22/2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-39* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04/25/2000* is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 9/22/2003.

As indicated in Applicant's response, claims 1,12, and 23 have been amended; and claims 34-39 added. Claims 1-39 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1-39 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-45 of U.S. Patent No. 6,631,516 (hereinafter '516).

Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations.

Following are but a few examples as to how the certain claims from the instant invention and from the above patent are conflicting with each other.

As per instant claims 1, 12, and 23, '516 claims 3, 18, and 33 also disclose, respectively, a method, system and apparatus for: receiving a programming language statements or user's specified commands wherein the statements or commands include attribute information declaring

symbol reference or definition; translating such programming statements into a object module including the symbol definition or symbol reference and attribute information for those symbol definition or reference; resolving the program object references to external symbol definition in another program.

But the above '516 claims do not explicitly disclose binding object modules with the attribute information available for binding; nor does it disclose producing an executable file. But the process of creating object files or modules and for resolving incompatibility of symbol references called from external program strongly suggests the process of binding. And it would have been obvious for one skill in the art to implement the attribute associating process from '516 to achieve the binding of objects as claimed in the instant claims in order to create the final executable for which all the attribute and symbol declaration have been generated because incompatibility checking of symbols is a necessary if not indispensable process for objects binding during the final translation into the executable code.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-7, 10-12, 14-18, 21-23, 25-29, and 35-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, USPN: 5,553,286 (hereinafter Lee), in view of Looney, USPN: 6,366,876 (hereinafter Looney) .

As per claim 1, Lee discloses a method, system (col. 4, lines 48-59; Fig. 1) of producing an executable file comprising:

receiving a plurality of programming language statements (e.g. col. 2, lines 3-7; Fig. 5, *TEXT2/TEXT*);

translating the source program (e.g. col. 1, lines 5-32; col. 4, lines 41-47; col. 5, line 58 to col. 6, line 6), including a symbol reference(*array of indexed data* – col. 6, lines 30-31); a symbol definition (Fig. 4, *section, class name*); attribute information for the symbol reference (*index numbers, size and offset fields* -- col. 6, line 51 to col. 7, line 1; *size, sequence, starting offset* – col. 8, lines 37-47), and attribute information for the symbol definition (*binding attribute* – col. 8, lines 37-46);

binding object modules into a program object, wherein the attribute information is available when so binding (e.g. col. 8, lines 37-54; col. 7, lines 4-30); and

resolving an external symbol reference (col. 8, lines 6-33; Fig. 5).

But, Lee does not expressly disclose that the attribute information are included in and declared within the program statements although Lee shows attributes helping the disposition of text segments in association with offset attributes while loading the module with a binder (e.g. Fig. 5; *class attributes* - col. 9, line 57 to col. 10, line 14). In a method to resolve incompatibilities and program symbol references analogous to Lee, Looney includes in the class additional declaration to provide attribute information for describing dependencies of program resources (e.g. program symbols or class definition) used in the symbol references resolution (e.g. *Attribute descriptors, Access Flags, Interface Descriptors* - col. 15, line 66 to col. 16, line 36; Figs 8A-C; *ConformRef 404* – Fig. 4). It would have been obvious for one of ordinary skill

in the art at the time the invention was made to modify the class definition as suggested by Lee in the binding process to resolve program text allocation correctness while loading the object module so that the declaration of attribute is provided within the class definition as taught by Looney, because this way programming resources dependencies and compatibilities can be resolved via direct parsing of program code resources, specifically when such compatibilities checking is for environment compatibility as taught by Looney (col. 1, line 49 to col. 2, line 47) and suggested by Lee's using of index and offset attributes to accommodate object module forming environment.

As per claim 3, Lee discloses that the object module is further capable of including fixed attribute information (e.g. RMODE - col. 7, lines 4-30, 47-49; *length, location* -- col. 9, lines 38-45; col. 3, lines 14-15).

As per claim 4, this claim is rejected for the same rationale set forth in claim 1 above, using Looney's method to generate additional attribute, or extending attribute information, for conflicts and compatibilities resolution.

As per claim 5, Lee discloses an address constant (*adcons* – col. 1, lines 48-60) for a symbol (Fig. 4, *section, class name)* and the attribute information declaring attribute information for the address constant (*class offsets, class identifier* -- col. 8, lines 48-52; *location and type of each address constant* - col. 3, lines 16-18).

As per claim 6, Lee discloses additional address constants for additional references to the symbol reference in the object module (*one or more address constants* -- col. 1, lines 56-61; col. 7, lines 41-45) and different attribute information sets for the address constants(e.g. col. 8, lines

50-54; *target segment, offsets, virtual address* -- col. 9, lines 6-14, 28-35, 54-67; col. 7, lines 1-3).

As per claim 7, Lee discloses that the resolving of the symbol reference and definition comprises a compatibility check *(Binder)* using the attribute information (*binding attribute, class identifier, offset* -- col. 8, lines 6-54; Fig. 5); and a separate compatibility checking for each reference to a symbol (col.6, line 51 to col. 7, line 1) for which there is a separate address constant and separate attribute information for each address constant (*target segment, offsets, virtual address* -- col. 9, lines 6-14, 28-35, 54-67).

As per claim 10, Lee discloses that the resolving of the symbol reference and definition comprises a compatibility check (see claim 7 above); but does not explicitly disclose that such resolving further comprises a compatibility check using signature data for the symbol definition and symbol reference. However, Looney in a method of assessing compatibility between programming resources analogous to that of Lee discloses the use of signature data per symbol reference (*MethodRef* – col. 12, lines 1-5) and symbol definition in the compatibility check within the symbol reference/definition resolution process (*signature attribute for method* -- col. 10, lines 3-42). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the signature data for symbol reference and definition just as taught by Looney into the method of compatibility check disclosed by Lee because this would further enforce the compliance of referenced data to be matched during the binding/linking process by virtue of the pre-determined signature data; thereby obviating extraneous usage of resources for recovery due to incompatibility errors.

As per claim 11, Lee does not expressly disclose the step of determining whether the compatibility check failed and the step of processing the attribute information declared for the symbol reference and definition that failed the compatibility check to determine a cause of the incompatibility. But Looney in an analogous method discloses the determining as to whether the symbol reference and definition compatibility check fail (Fig. 9a-c); and the processing of the attribute information (e.g. *compliance status* – col. 10, lines 20-64*)* for such symbol reference and definition(*class, method, fields, return type*) to determine the cause of incompatibility(e.g. col. 2, lines 19-47; col. 2 line 59 to col. 3, line 6; col. 10, lines 20-64; col. 12, lines 54-58). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the compatibility failure checking for symbol reference and definition; and determining of cause thereof such as taught by Looney into the method of compatibility check disclosed by Lee. The modification would have been obvious because this would further establish a systematic recording of the compliance checking results on referenced data during the binding/linking process, thereby providing a base of information for the analysis and/or improvement of future incompatibility error checking processes.

As per claims 12, 14 and 16, these claims recite a system comprising the same corresponding limitations set forth in claims 1, 2 and 5 above, respectively. Hence, the rejections of claims 1, 2, and 5 are herein applied.

As per claims 15 and 26, these are the system and apparatus versions of claim 4 above; and are rejected with the same rationale.

As per claims 17 and 18, these are the system versions of the methods in respectively, claims 6 and 7 above; whose limitations have already been addressed above.

As per claims 21, 22 and 32, 33, these are respectively the system and computer product versions of the methods in respectively, claims 10 and 11 above. Hence, the rejections of claims 10 and 11, respectively, are herein applied.

As per claim 23, 25 and 27, these claims recite an article of manufacturer comprising a computer usable media embedding a computer program capable of performing the same method steps in claims 1, 2, and 5 respectively, which limitations have been addressed in the rejection of claims 1, 2, and 5, respectively. Furthermore, Lee also teaches such computer product/article of manufacturer in col. 10, line 66 to col. 12, line 9.

As per claims 28 and 29, these are the computer product versions of the methods in respectively, claims 6 and 7 above; whose limitations have already been addressed above.

As per claims 34, Lee discloses class attributes, i.e. attributes to symbol definition, and attribute information for support of the binding of program text into the module (re claim 1) but does not specify that such attribute information describes whether symbol definition or reference is to data or to executable code. But Looney teaches enhancing the program language with additional specifications such as including attributes in reference classes to enable improved resolution of symbol definition or references or environment incompatibilities, thus declaring attributes applying to data types or interdependent classes, methods or package definition analogous to executable resources (e.g. Figs. 4, 7-9). It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the class attribute and definition as suggested by Lee in the binding process to resolve program text allocation correctness while loading the object module so that the declaration of attribute is provided within the class definition and extended specification informing of data type and executable resources

as taught by Looney, because this way programming resources dependencies and compatibilities, in terms of both data and code, can be resolved via direct parsing of program code resources, specifically when such compatibilities are for environment compatibility as taught by Looney (col. 1, line 49 to col. 2, line 47) and suggested by Lee's using of index and offset attributes to accommodate object module forming environment.

As per claims 35, Looney disclose extension of class specification as to include attribute information describing parameters passed when applied to methods and Type when applied to data (e.g. Fig. 8A) while Lee only discloses attributes externally created to support binding of class text and object modules (re claim 1). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the extension of class definition and declaration by Lee the implementing of Looney's extended class specification with attribute describing data type or method parameters so to provide enhanced programming resources definition or reference conflict resolution in order to maintain a binding of code free of incompatibilities when not only code but also data are subjected to conflicts as suggested by Looney from above.

As per claims 36 and 38, these are the system and apparatus versions of claim 34; hence are rejected as have been addressed therein.

As per claims 37 and 39, these are the system and apparatus versions of claim 35; hence are rejected as have been addressed therein

6. Claims 2, 8, 9, 13, 19, 20, 24, 30, and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, USPN: 5,553,286, and Looney, USPN: 6,366,876, as applied to claims 1, 12, 23 above, and further in view of Fitzgerald, USPN: 5,408,665 (hereinafter Fitzgerald).

As per claim 2, Lee does not explicitly disclose that the language statement is capable of indirectly declaring extended attribute information defined in another location in the object module. Lee, however, discloses external symbol storage and relocation dictionary (col. 2, lines 1-9) and Looney teaches class attribute declaration within a class to inform about externally declared classes (e.g. *package 816: parent* – Fig. 8B; *ParentClass 412* - Fig. 4 – Note: declaration of class is equivalent to declaring a reference object). Further, Fitzgerald, in a method to bind object modules translated from program source into executables using external symbol resolution analogous to that of Lee, discloses attribute information indirectly defined in another location of the object module attribute, i.e. MODULE ID, such as to set a link via a pointer in the external dictionary to get to an attribute BLOCK ID declared externally to the module in which MODULE ID has been referenced (*Module ID* – Fig. 4b, 5b, 6B,C, D). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the indirect definition of an attribute via the use of a pointer as reference to a externally defined attribute information as taught by Fitzgerald to the method of using external symbol resolution disclosed by Lee combined with the method of declaring reference classes and additional attribute for external reference by Looney. The modification would have been obvious because the usage of reference classes by Looney would be facilitated via indirect referencing as taught by Fitzgerald while using the advantage of class objects and object-oriented non-concurrence properties, thus improve the time and direct storage resource usage efficiency of the modules linking and binding process in Lee's disclosed system, such storage efficiency being further enhanced by indirect reference just as suggested by Fitzgerald to alleviate cumbersome overloading of data reference at one address location.

As per claim 8, Lee further discloses that the object module includes an External Symbol Directory (ESD) including a record capable of indicating a symbol in the program, a location of the symbol in the program (col. 6, lines 54-57; Fig. 5, *ESD 64*); but does not point out that such ESD includes a pointer to attribute information in the program for the symbol. However, Looney discloses declaring reference classes to refer to externally declared symbols or program resources (re claim 2). Further, Fitzgerald in a method to bind object modules translated from program source into executables analogous to that of Lee discloses the use of pointer in an external symbol table (*External Dictionary*) to refer to the attribute information in the program for the symbol (e.g. Fig. 4b, 5b, 6B,C, D). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the implementation of pointer to symbol attribute inside the ESD as taught by Fitzgerald into Lee's ESD combined with Looney's use of class reference with extended attributes because such additional pointing structure or class reference would facilitate the fetching of attribute needed to resolve symbol references during binding of modules components, alleviate the data (e.g. whole attribute data) store usage in the ESD by just storing an information object referring to that data, and further improve efficiency by using object-oriented properties such as non-concurrency of declared objects.

As per claim 9, Lee discloses that the object module further includes an Relocation List Directory (RLD), a location of an address constant (col. 3, lines 16-18; col. 8, lines 48-54; Fig. 5 – *RLD 64*); but does not point out that such RLD includes a pointer to attribute information for the address constant in the program. However, Looney discloses declaring reference classes to refer to externally declared symbols or program resources (re claim 2). Fitzgerald, in a method to bind object modules translated from program source into executables analogous to that of Lee

discloses the use of external tables (e.g. step *603*-Fig. 6A; *external list* – Fig. 5B); pointer in a symbol dictionary (*STD. DICT.*) and External Dictionary to point to the second attribute information (e.g. in form of an identification string) indirectly defining the first referenced program module attribute (Fig. 4b, 5b, 6B,C, D; *MODULE ID* → *PTR* (*Ext. Dict*) → *ASCII STRING, BUCKET*- Fig. 5B). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the implementation of pointer to address constant inside an external table or symbol directory, i.e. relocating tables, as taught by Fitzgerald into Lee's RLD referencing technique in light of Looney's use of reference class declared in the program because such additional pointing structure (a class being equivalent to a pointer object) would facilitate the fetching of attribute needed to resolve symbol references during binding of modules components, alleviate the data (e.g. whole attribute data) store usage in the ESD by just storing an address referring to that data; and improve efficiency in using object-oriented properties such as non-concurrency of declared objects.

As per claims 13, 19, 20, and 24, 30, 31,these claims are respectively the system and computer product versions of the method limitations set forth and addressed respectively, in claims 2, 8, and 9 above. Hence, the rejections of claims 2, 8 and 9, respectively, are herein applied.

Response to Arguments

7. Applicant's arguments filed 9/22/2003 have been fully considered but they are moot in view of new grounds of rejection. Following are the reasons therefor.

(A) As per claims 1, 12, 23, arguments by Applicants (Applicants' Remarks, pg. 10 to pg. 13,top), the deficiency of Lee in regard to the newly amended claim limitation as observed by

Applicants is currently remedied by Looney's use program declaration of extended class references and attributes to symbol definition or other program resource reference; thus are moot in view of the new grounds of rejection.

(B) As per claims 3, 5-7, 14, 16-18, 25, and 27-29, arguments by Applicants (Applicants' Remarks, pg. 13, bottom, pg. 14, and top pg. 15) concerning Lee's deficiency in disclosing attribute information derived from language statements are also moot in view of the current grounds of rejection for the same reasons just mentioned.

(C) As per claims 2, 4, 8, 9, 13, 15, 19-20, 24, 26, 30 and 31, arguments by Applicants (Applicants' Remarks, pg. 16-17) that Fitzgerald lacks pointer to attribute information for a symbol definition or reference as claimed; and that Fitzgerald does not disclose attribute information derived from program statements are now moot because the current rejection is using Looney to address program declaration of attribute information and Fitzgerald to provide the use of pointer to a indirectly declared attribute information, with both teaching being used in providing the current motivation for combination. Attacking Fitzgerald or Lee separately would have been inapposite in view of the new combination as set forth in the rejection.

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

U.S. Pat No. 6,185,571 to Carter et al., disclosing extending language to include additional attribute for data specification to resolve data incompatibilities.

Applicant's arguments filed 9/22/2003 have been fully considered but they are moot in view of new grounds of rejection.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please label "PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

VAT
November 23, 2003

TODD INGBERG
PRIMARY EXAMINER